Architectural Decisions Document

ECG Heartbeat Classification With Images

This project focuses on accurately classifying five different rhythms:

- Normal
- Atrial Premature Contraction
- Premature Ventricular Contraction
- Fusion of Ventricular and Normal Contractions
- Paced, Unclassifiable heartbeats

1.1 Data Source

The initial dataset is composed of two collections of heartbeat signals derived from two datasets in heartbeat classification: MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. The dataset is also available through Kaggle. Dataset Link: https://www.kaggle.com/shayanfazeli/heartbeat?select=mitbih_train.csv

1.1.1 Technology Choice

Dataset contained in CSV file format

1.1.2 Justification

CSV files are easier for the end user and can be expanded as more data becomes available.

1.2 Enterprise Data

Not applicable.

1.2.1 Technology Choice

Not applicable.

1.2.2 Justification

Project does not use enterprise data.

1.3 Streaming analytics

Not applicable.

1.3.1 Technology Choice

Not applicable.

1.3.2 Justification

Project does not use streaming data.

1.4 Data Integration

The MIT-BIH Arrhythmia Dataset from Kaggle was examined to Identify quality issues, assess features, and understand the value distributions of the data using statistical measures and visualizations. All ECG samples were then plotted, converted into jpg images, stored in a csv file and finally joined with a csv file of the diagnostic labels to create a new dataset suitable for use downstream in the project.

1.4.1 Technology Choice

Python Pandas, numpy, matplotlib and OpenCV are used in the data exploration and transformation of the initial dataset and creation of the new dataset used in this project.

1.4.2 Justification

Pandas, NumPy and Matplotlib and OpenCV are widely used, open-source libraries with extensive documentation and community support.

1.5 Data Repository

Persistent storage is used for this project.

1.5.1 Technology Choice

Cloud object storage.

1.5.2 Justification

1. Object storage is the least expensive option for storage.
2. Any data type is supported.
3. Scales to the petabyte range.
4. Can access specific storage locations through folder and file names and file offsets.

1.6 Discovery and Exploration

Statistical analysis and visualizations of ECG samples are used to examine ECG signals, numeric and categorical distributions and correlations of the dataset features.

1.6.1 Technology Choice

Pandas, NumPy, Matplotlib and OpenCV are used for analysis and visualizations.

1.6.2 Justification

Pandas, NumPy, Matplotlib and OpenCV are widely used, open-source libraries with extensive documentation and community support.

1.7 Actionable Insights

This project is a multiclass classification task. The metric is Accuracy along with Precision, Recall/Sensitivity, and Specificity.

Algorithms

Non-Deep Learning:

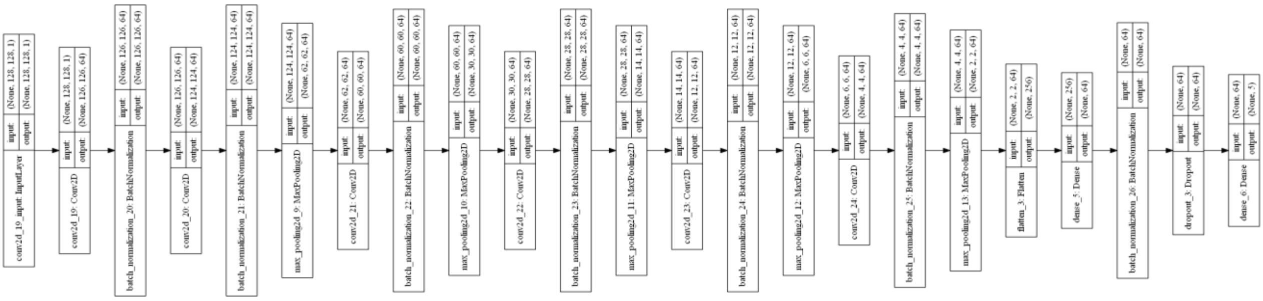
- Decision Tree
- Random Forest
- XGBoost Classifier
- K-Nearest Neighbors Classifier

Neural Network Architectures:

- Single fully connected layer with same number of neurons as the input variables, one hidden layer with 512 neurons and a softmax output layer.
- Second model is the same, but with an additional hidden layer of 64 neurons and a dropout layer at (0.2)

2D Convolutional Neural Networks:

- Model 1 trained using 28 x 28 sized images
- Model 2 trained using 128 x 128 sized images


conv2d_19_input: InputLayer
input: (None, 128, 128, 1)
output: (None, 128, 128, 1)
conv2d_19: Conv2D
input: (None, 128, 128, 1)
output: (None, 126, 126, 64)
batch_normalization_20: BatchNormalization
input: (None, 126, 126, 64)
output: (None, 126, 126, 64)
conv2d_20: Conv2D
input: (None, 126, 126, 64)
output: (None, 124, 124, 64)
batch_normalization_21: BatchNormalization
input: (None, 124, 124, 64)
output: (None, 124, 124, 64)
max_pooling2d_9: MaxPooling2D
input: (None, 124, 124, 64)
output: (None, 62, 62, 64)
conv2d_21: Conv2D
input: (None, 62, 62, 64)
output: (None, 60, 60, 64)
batch_normalization_22: BatchNormalization
input: (None, 60, 60, 64)
output: (None, 60, 60, 64)
max_pooling2d_10: MaxPooling2D
input: (None, 60, 60, 64)
output: (None, 30, 30, 64)
conv2d_22: Conv2D
input: (None, 30, 30, 64)
output: (None, 28, 28, 64)
batch_normalization_23: BatchNormalization
input: (None, 28, 28, 64)
output: (None, 28, 28, 64)
max_pooling2d_11: MaxPooling2D
input: (None, 28, 28, 64)
output: (None, 14, 14, 64)
conv2d_23: Conv2D
input: (None, 14, 14, 64)
output: (None, 12, 12, 64)
batch_normalization_24: BatchNormalization
input: (None, 12, 12, 64)
output: (None, 12, 12, 64)
max_pooling2d_12: MaxPooling2D
input: (None, 12, 12, 64)
output: (None, 6, 6, 64)
conv2d_24: Conv2D
input: (None, 6, 6, 64)
output: (None, 4, 4, 64)
batch_normalization_25: BatchNormalization
input: (None, 4, 4, 64)
output: (None, 4, 4, 64)
max_pooling2d_13: MaxPooling2D
input: (None, 4, 4, 64)
output: (None, 2, 2, 64)
flatten_3: Flatten
input: (None, 2, 2, 64)
output: (None, 256)
dense_5: Dense
input: (None, 256)
output: (None, 64)
batch_normalization_26: BatchNormalization
input: (None, 64)
output: (None, 64)
dropout_3: Dropout
input: (None, 64)
output: (None, 64)
dense_6: Dense
input: (None, 64)
output: (None, 5)

1.7.1 Technology Choice

Python with libraries (pandas, scikit-learn, numpy, OpenCV, tensorflow and keras) are used with IBM Watson Studio and pre-run Jupyter notebooks with explanations of each step.

1.7.2 Justification

The project components are open source with extensive documentation and community support. Scikit-learn, TensorFlow and Keras support a wide range of state-of-the-art models.

TensorFlow is one of the most widely used deep learning frameworks. At its core, it is a linear algebra library supporting automatic differentiation. Keras provides an abstraction layer on top of TensorFlow.

Model Selection

Decision tree - The process behind the model is easy to understand and clear when viewing a plot of the model. Only the important attributes in making a decision are included in its rule. Attributes that do not contribute to the accuracy of the model are ignored.

Random forests - operate by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Random forests correct for individual decision tree overfitting.

XGBoost - Extreme Gradient Boosting is a supervised leaning gradient boosted trees algorithm. The gradient boosting technique attempts to accurately predict targets by combining the estimates of simpler, weaker models.

K-Nearest Neighbours - The KNN algorithm is suitable for applications that require high accuracy but that do not require a human-readable model.

Neural Network - Neural Networks can recognize patterns and produce output that is not limited to the input provided to them. Neural networks learn from examples and can apply them in similar situations.

2D Convolutional Neural Networks – Convnets are deep-learning models used almost universally in computer vision applications. These models can yield reasonable results when trained on small datasets without the need for custom feature engineering.

Performance Metrics

Accuracy is the proportion of correct predictions over the total number of predictions.
Accuracy = (TP + TN) / All Predictions

Precision (Positive Predictive Value - PPV), out of all predicted positive cases, how many were actually positive.
Precision = TP / (TP + FP)

Sensitivity (Recall), out of all actual positives how many were predicted as positive.
Sensitivity (Recall) = TP / (FN + TP)

Specificity (Selectivity or True Negative Rate – TNR), out of all actual negatives (not a certain lesion), how many were predicted as negative.
Specificity = TN / (TN + FP)

Out of the various combinations of Precision, Recall/Sensitivity, and Specificity applied to algorithms we are looking for those where we have **High Precision, High Recall/Sensitivity**, and **High Specificity**.

1.8 Applications / Data Products

1.8.1 Technology Choice

Jupyter notebook and trained model with explanation of the entire process.
Trained model and weights.

1.8.2 Justification

A Jupyter notebook with a trained model and explanation is simple to use for testing in a local environment.
The model could be used to classify ECG signals as an augmentation of diagnoses.

1.9 Security, Information Governance and Systems Management

1.9.1 Technology Choice

Project Jupyter notebooks and saved model stored on the IBM Cloud Platform and GitHub repository. Data files in cloud object storage

1.9.2 Justification

Security for these low or no cost services is managed by the vendors.